下载APP

01｜前世今生：你不得不了解的Go的历史和现状

2021-10-13 Tony Bai

《Tony Bai · Go语言第一课》 课程介绍 >

讲述：Tony Bai
时长 14:27 大小 13.24M

你好，我是 Tony Bai。

今天是我们的第一堂课。第一堂课的开场，我要做的事很简单，就想跟你聊一聊 Go 语言的前世今生。

我一直认为，当你开始接触一门新语言的时候，你一定要去了解它的历史和现状。因为这样，你才能建立起对这门语言的整体认知，了解它未来的走向。而且，也能建立你学习的“安全感”，相信它能够给你带来足够的价值和收益，更加坚定地学习下去。

所以，在这一节课，我就来跟你聊聊 Go 的前世今生，讲清楚 Go 到底是一门怎么样的语言，Go 又是怎么诞生的，它经历了怎样的历史演进，它的现状和未来又会如何？

无论后面你是否会选择学习 Go 语言，无论你是否会真正成为一名是 Go 程序员，我都建议你先了解一下这些内容，它会让你对编程语言的发展有更进一步的理解。

首先，我们就来看看 Go 语言是怎么诞生的，这可以让你真实地了解 Go 的诞生缘由、设计目标，以及它究竟要解决哪些问题。

Go 语言是怎样诞生的？

Go 语言的创始人有三位，分别是图灵奖获得者、C 语法联合发明人、Unix 之父肯·汤普森（Ken Thompson）、Plan 9 操作系统领导者、UTF-8 编码的最初设计者罗伯·派克（Rob Pike），以及 Java 的 HotSpot 虚拟机和 Chrome 浏览器的 JavaScript V8 引擎的设计者之一罗伯特·格瑞史莫（Robert Griesemer）。

他们可能都没有想到，他们三个人在 2007 年 9 月 20 日下午的一次普通讨论，就这么成为了计算机编程语言领域的一次著名历史事件，开启了一个新编程语言的历史。

Go语言之父们（从左到右分别是Robert Griesemer、Rob Pike和Ken Thompson）

那天下午，在谷歌山景城总部的那间办公室里，罗伯·派克启动了一个 C++ 工程的编译构建。按照以往的经验判断，这次构建大约需要一个小时。利用这段时间，罗伯·派克和罗伯特·格瑞史莫、肯·汤普森坐在一处，交换了关于设计一门新编程语言的想法。

之所以有这种想法，是因为**当时的谷歌内部主要使用 C++ 语言构建各种系统，但 C++ 的巨大复杂性、编译构建速度慢以及在编写服务端程序时对并发支持的不足**，让三位大佬感觉到十分不便，他们就想着设计一门新的语言。在他们的初步构想中，这门新语言应该是能够给程序员带来快乐、匹配未来硬件发展趋势并适合用来开发谷歌内部大规模网络服务程序的。

趁热打铁！在第一天的简短讨论后，第二天这三位大佬又在谷歌总部的“雅温得（Yaounde）”的会议室里具体讨论了这门新语言的设计。会后罗伯特·格瑞史莫发出了一封题为“prog lang discussion”的电邮，对这门新编程语言的功能特性做了初步的归纳总结：

```
Date: Sun, 23 Sep 2007 23:33:41 -0700
From: "Robert Griesemer" <gri@google.com>
To: "Rob 'Commander' Pike" <r@google.com>, ken@google.com
Subject: prog lang discussion
...
*** General:
Starting point: C, fix some obvious flaws, remove crud, add a few missing features
  - no includes, instead: import
  - no macros (do we need something instead?)
  - ideally only one file instead of a .h and .c file, module interface
should be extracted automatically
  - statements: like in C, though should fix 'switch' statement
  - expressions: like in C, though with caveats (do we need ',' expressions?)
  - essentially strongly typed, but probably w/ support for runtime types
  - want arrays with bounds checking on always (except perhaps in 'unsafe mode'-see section on GC)
  - mechanism to hook up GC (I think that most code can live w/ GC, but for a true systems
    programming language there should be mode w/ full control over memory allocation)
  - support for interfaces (differentiate between concrete, or implementation types, and abstract,
    or interface types)
  - support for nested and anonymous functions/closures (don't pay if not used)
  - a simple compiler should be able to generate decent code
  - the various language mechanisms should result in predictable code
...
```

Go语言第一版特性设计稿

这封电邮对这门新编程语言的功能特性做了归纳总结。主要思路是，**在 C 语言的基础上，修正一些明显的缺陷，删除一些被诟病较多的特性，增加一些缺失的功能**，比如，使用 import 替代 include、去掉宏、增加垃圾回收、支持接口等。这封电邮成为了这门新语言的第一版特性设计稿，三位大佬在这门语言的一些基础语法特性上达成了初步一致。

9 月 25 日，罗伯·派克在一封回复电邮中把这门新编程语言命名为**“go”**：

Subject: Re: prog lang discussion
From: Rob 'Commander' Pike
Date: Tue, Sep 25, 2007 at 3:12 PM
To: Robert Griesemer, Ken Thompson

i had a couple of thoughts on the drive home.

1. name

'go'. you can invent reasons for this name but it has nice properties. it's short, easy to type. tools: goc, gol, goa. if there's an interactive debugger/interpreter it could just be called 'go'. the suffix is .go
...

新编程语言被命名为“go”

在罗伯·派克的心目中，“go”这个单词短小、容易输入并且在组合其他字母后便可以用来命名 Go 相关的工具，比如编译器（goc）、汇编器（goa）、链接器（gol）等（go 的早期版本曾如此命名 go 工具链，但后续版本撤销了这种命名方式，仅保留 go 这一统一的工具链名称 ）。

这里我还想澄清一个误区，很多 Go 语言初学者经常称这门语言为 golang，其实这是不对的：“golang”仅应用于命名 Go 语言官方网站，而且当时没有用 go.com 纯粹是这个域名被占用了而已。

从“三人行”到“众人拾柴”

经过早期讨论，Go 语言的三位作者在语言设计上达成初步一致后，便开启了 Go 语言迭代设计和实现的过程。

2008 年初，Unix 之父肯·汤普森实现了第一版 Go 编译器，用于验证之前的设计。这个编译器先将 Go 代码转换为 C 代码，再由 C 编译器编译成二进制文件。

到 2008 年年中，Go 的第一版设计就基本结束了。这时，同样在谷歌工作的伊恩·泰勒（Ian Lance Taylor）为 Go 语言实现了一个 gcc 的前端，这也是 Go 语言的第二个编译器。

伊恩·泰勒的这一成果不仅仅是一种鼓励，也证明了 Go 这一新语言的可行性 。有了语言的第二个实现，对 Go 的语言规范和标准库的建立也是很重要的。随后，伊恩·泰勒以团队的第四位成员的身份正式加入 Go 语言开发团队，后面也成为了 Go 语言，以及其工具设计和实现的核心人物之一。

罗斯·考克斯（Russ Cox）是 Go 核心开发团队的第五位成员，也是在 2008 年加入的。进入团队后，罗斯·考克斯利用函数类型是“一等公民”，而且它也可以拥有自己的方法这个特性巧妙设计出了 http 包的`HandlerFunc`类型。这样，我们通过显式转型就可以让一个普通函数成为满足`http.Handler`接口的类型了。

不仅如此，罗斯·考克斯还在当时设计的基础上提出了一些更泛化的想法，比如`io.Reader`和`io.Writer`接口，这就奠定了 Go 语言的 I/O 结构模型。后来，罗斯·考克斯成为 Go 核心技术团队的负责人，推动 Go 语言的持续演化。

到这里，Go 语言最初的核心团队形成，Go 语言迈上了稳定演化的道路。

2009 年 10 月 30 日，罗伯·派克在 Google Techtalk 上做了一次有关 Go 语言的演讲“The Go Programming Language”，这也是 Go 语言第一次公之于众。十天后，也就是 2009 年 11 月 10 日，谷歌官方宣布 Go 语言项目开源，之后这一天也被 Go 官方确定为 Go 语言的诞生日。

Hey! Ho! Let's Go!

Tuesday, November 10, 2009

Here at Google, we believe programming should be fast, productive, and most importantly, fun. That's why we're excited to open source an experimental new language called Go. Go combines the development speed of working in a dynamic language like Python with the performance and safety of a compiled language like C or C++. Typical builds feel instantaneous; even large binaries compile in just a few seconds. And the compiled code runs close to the speed of C. Go lets you move fast.

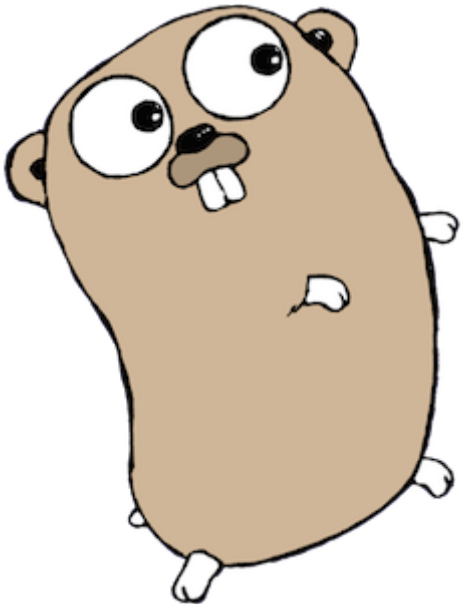
Go is a great language for systems programming with support for multi-processing, a fresh and lightweight take on object-oriented design, plus some cool features like true closures and reflection.

Want to write a server with thousands of communicating threads? Want to spend less time reading blogs while waiting for builds? Feel like whipping up a prototype of your latest idea? Go is the way to go! Check out the video for more information or visit golang.org.

By Robert Griesemer, Rob Pike, Ken Thompson, Ian Taylor, Russ Cox, Jini Kim and Adam Langley - The Go Team

谷歌宣布Go语言项目开源

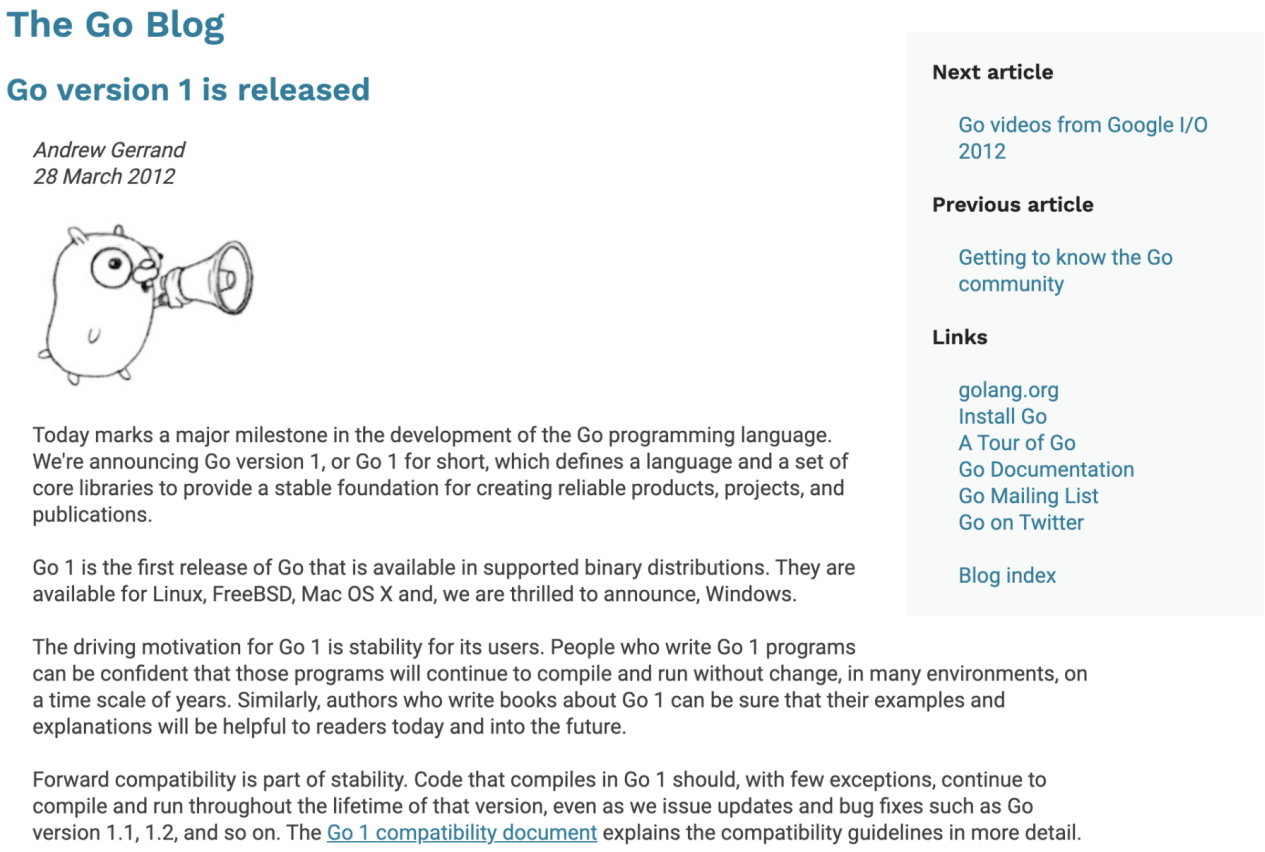
在 Go 语言项目开源后，Go 语言也迎来了自己的“吉祥物”，是一只由罗伯·派克夫人芮妮·弗伦奇（Renee French）设计的地鼠，从此地鼠（gopher）也就成为了世界各地 Go 程序员的象征，Go 程序员也被昵称为 **Gopher**，在后面的课程中，我会直接使用 Gopher 指代 Go 语言开发者。



Go语言的吉祥物

Go 语言项目的开源使得 Go 语言吸引了全世界开发者的目光，再加上 Go 三位作者在业界的影响力以及谷歌这座大树的加持，更多有才华的程序员加入到 Go 核心开发团队中，更多贡献者开始为 Go 语言项目添砖加瓦。于是，Go 在宣布开源的当年，也就是 2009 年，就成为了著名编程语言排行榜 TIOBE 的年度最佳编程语言。

2012 年 3 月 28 日，Go 1.0 版本正式发布，同时 Go 官方发布了“Go1 兼容性”承诺：**只要符合 Go1 语言规范的源代码，Go 编译器将保证向后兼容（backwards compatible），也就是说我们使用新版编译器也可以正确编译用老版本语法编写的代码。**



The Go Blog

Go version 1 is released

Andrew Gerrand
28 March 2012

Today marks a major milestone in the development of the Go programming language. We're announcing Go version 1, or Go 1 for short, which defines a language and a set of core libraries to provide a stable foundation for creating reliable products, projects, and publications.

Go 1 is the first release of Go that is available in supported binary distributions. They are available for Linux, FreeBSD, Mac OS X and, we are thrilled to announce, Windows.

The driving motivation for Go 1 is stability for its users. People who write Go 1 programs can be confident that those programs will continue to compile and run without change, in many environments, on a time scale of years. Similarly, authors who write books about Go 1 can be sure that their examples and explanations will be helpful to readers today and into the future.

Forward compatibility is part of stability. Code that compiles in Go 1 should, with few exceptions, continue to compile and run throughout the lifetime of that version, even as we issue updates and bug fixes such as Go version 1.1, 1.2, and so on. The Go 1 compatibility document explains the compatibility guidelines in more detail.

Next article

Go videos from Google I/O 2012

Previous article

Getting to know the Go community

Links

golang.org
Install Go
A Tour of Go
Go Documentation
Go Mailing List
Go on Twitter

Blog index

Go 1.0版本正式发布

从此，Go 语言发展得非常迅猛。从正式开源到现在，十一年的时间过去了，Go 语言发布了多个大版本更新，逐渐成熟。这里，我也梳理了迄今为止 Go 语言的重大版本更新，希望能帮助你快速了解 Go 语言的演化历史。

Go语言大事记

Go 是否值得我们学习？

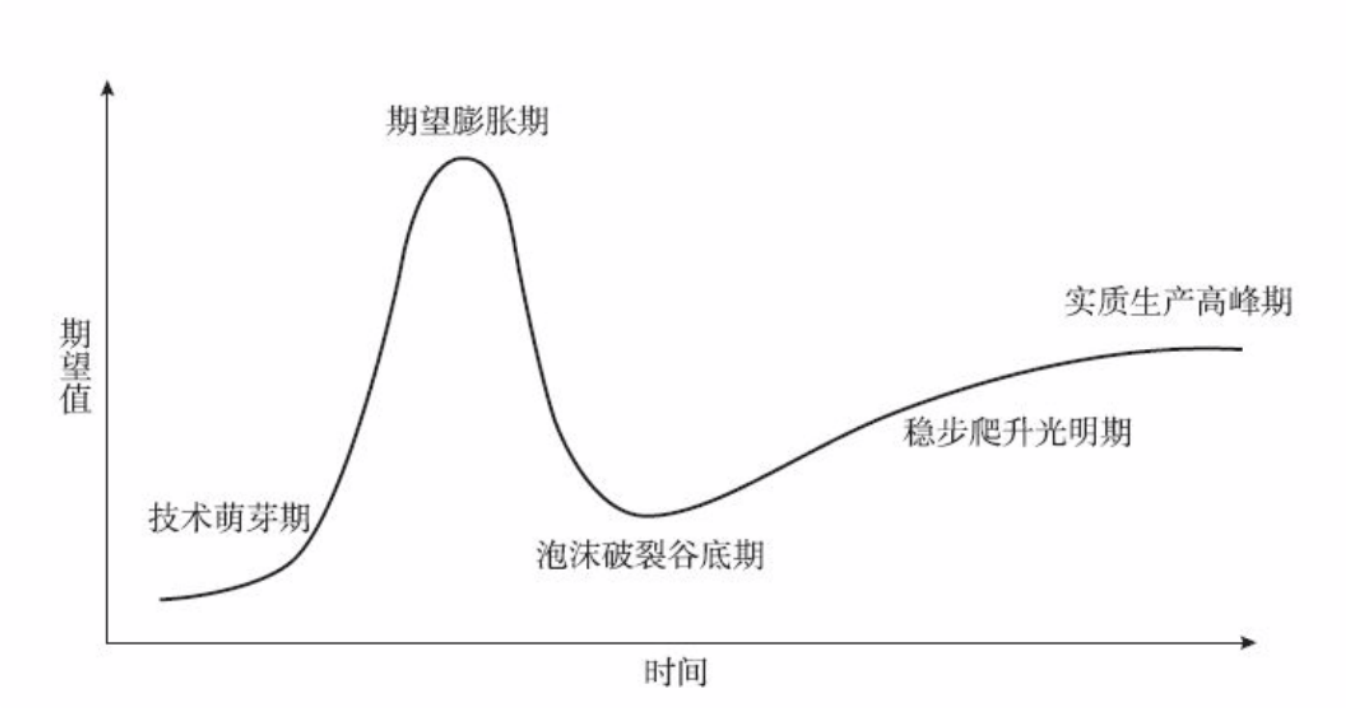
时间已经来到了 2021 年。经过了十余年的打磨与优化，如今的 Go 语言已经逐渐成为了云计算时代基础设施的编程语言。你能想到的现代云计算基础设施软件的大部分流行和可靠的作品，都是用 Go 编写的，比如：Docker、Kubernetes、Prometheus、Ethereum（以太坊）、Istio、CockroachDB、InfluxDB、Terraform、Etcd、Consul 等等。当然，这个列表还在持续增加，可见 Go 语言的影响力已经十分强大。

Go 除了在云计算基础设施领域，拥有上面这些杀手级应用之外，Go 语言的用户数量也在近几年快速增加。Go 语言项目技术负责人罗斯·考克斯甚至还专门写过一篇文章，来估算全世界范围的 Gopher 数量。按照他的估算结果，全世界范围的 Gopher 数量从 2017 年中旬的最多 100 万，增长到 2019 年 11 月的最多 196 万，大概每 18 个月翻一番。庞大的 Gopher 基数为 Go 未来的发展提供持续的增长潜力和更大的想象空间。

那么 Go 语言前景究竟如何，值不值得投入去学习呢？

我在想，是否存在一种成熟的方法，能相对客观地描绘出 Go 语言的历史发展趋势，并对未来 Go 的走势做出指导呢？我想来想去，觉得 Gartner 的技术成熟度曲线（The Hype Cycle）可以借来一试。

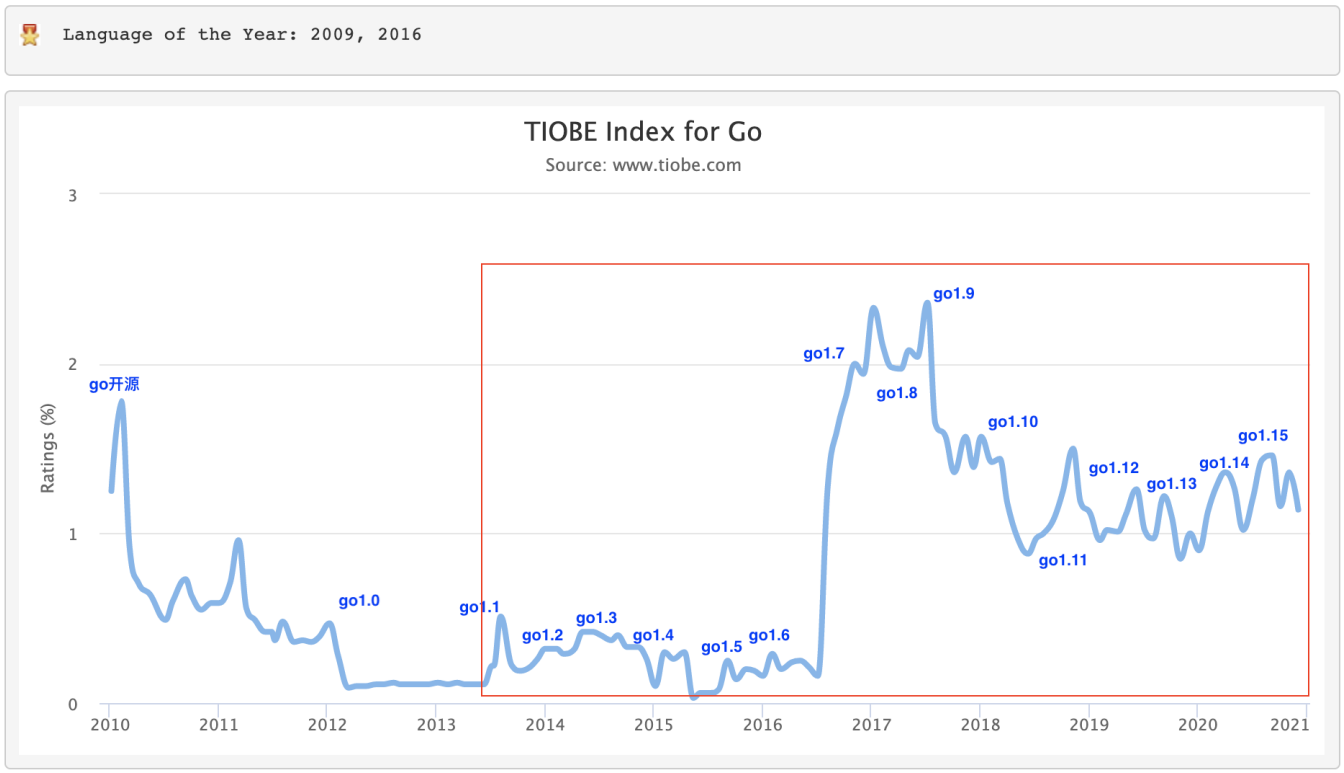
Gartner 的技术成熟度曲线又叫技术循环曲线，是企业用来评估新科技是否要采用或采用时机的一种可视化方法，它利用时间轴与该技术在市面上的可见度（媒体曝光度）决定要不要采用，以及什么时候采用这种新科技，下面就是一条典型的技术成熟度曲线的形状：



Gartner技术成熟度曲线

同理，如果我们将这条技术成熟度曲线应用于某种编程语言，比如 Go，我们就可以用它来判断这门编程语言所处的成熟阶段，来辅助我们决定要不要采用，以及何时采用这门语言。

我们从知名的 TIOBE 编程语言指数排行榜获取 Go 从 2009 年开源以来至今的指数曲线图，并且根据 Go 版本发布历史在图中标记出了各个时段的 Go 发布版本，你可以看看。



Go语言的技术成熟度曲线

对比前面的 Gartner 成熟度曲线，我们可以得出这样的结论：Go 在经历了一个漫长的技术萌芽期后，从实现自举的 Go 1.5 版本开始逐步进入“期望膨胀期”，在经历从 Go 1.6 到 Go 1.9 版本的发布后，业界对 Go 的期望达到了峰值。

但随后“泡沫破裂”，在 Go 1.11 发布前跌到了“泡沫破裂期”的谷底，Go 1.11 版本引入了 Go module，给社区解决 Go 包依赖问题注射了一支强心剂，于是 Go 又开始了缓慢爬升。

从 TIOBE 提供的曲线来看，Go 1.12 到 Go 1.15 版本的发布让我们有信心认为 Go 已经走出“泡沫破裂谷底期”，进入到“稳步爬升的光明期”。

至于 Go 什么时候能达到实质生产高峰期呢？

我们还不好预测，但这应该是一个确定性事件。我认为现在离它到达实质生产高峰期只是一个时间问题了。也许预计在 2022 年初发布的支持 Go 泛型特性的 Go 1.18 版本，会是继 Go 1.5 版本之后又一“爆款”，很可能会快速推动 Go 迈入更高的发展阶段。

小结

到这里，我们今天这节课就结束了。在这一节课里，我们一起探讨了**“Go 从哪里来，并可能要往哪里去”**的问题。

我前面也说了，一门编程语言的历史和现状，能给你带来学习的“安全感”，相信它可以提升你的个人价值，也会让你获得丰厚的回报。你也会更加清楚地认识到：自己为什么要学习它？它未来的发展趋势又是怎样的？而且，当这门语言的现状能给予你极大“安全感”的时候，我们才会“死心塌地”地学习和钻研这门语言，而不会有太多的后顾之忧。

从 Go 本身的发展来看，和多数编程语言一样，Go 语言在诞生后，度过了一个较长的“技术萌芽期”。然后，实现了自举，而且对 GC 延迟进行了大幅优化的 Go 1.5 版本，成为了 Go 语言演化过程中的第一个“引爆点”，推动 Go 语言进入“技术膨胀期”。

也正是在这段时间内，Go 语言以迅雷不及掩耳盗铃之势推出了以 Docker、Kubernetes 为典型代表的“杀手级应用”，充分展现了实力，在世界范围收获了百万粉丝，迸发出极高的潜力和持续的活力。

Go 开源于 2009 年末，如果从那时算起，Go 才 11 岁。但在 Go 核心开发团队眼中，Go 的真正诞生年份是 2007 年，距今已 13 个年头有余了。

回顾一下计算机编程语言的历史，我们会发现，绝大多数主流编程语言，都将在其 15 至 20 年间大步前进。Java、Python、Ruby、JavaScript 和许多其他编程语言都是这样。如今 Go 语言也马上进入自己的黄金 5～10 年，从前面的技术成熟度曲线分析也可以印证这一点：Go 已经重新回到“稳步爬升的光明期”。

对于开发人员来说，**Go 语言学习的最佳时刻已经到来了**！

思考题

相较于传统的静态编译型编程语言（如 C、C++），Go 做出了哪些改进？你可以思考一下，欢迎在留言区留下你的答案。

感谢你和我一起学习，也欢迎你把这节课分享给更多对 Go 语言感兴趣的朋友。我是 Tony Bai，我们下节课见。

分享给需要的人，Ta订阅后你可得 20 元现金奖励

生成海报并分享

赞 11 提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 开篇词 | 这样入门Go，才能少走弯路

下一篇 02 | 拒绝“Hello and Bye”：Go语言的设计哲学是怎么一回事？

精选留言 (9)

写留言

lesserror

2021-10-13

Tony Bai，你好。文章读完了，写的很好。关于文中有几点疑问，麻烦解答一下。

1. “为 Go 语言实现了一个 gcc 的前端？” 怎么理解这个“前端”的意思。

2. “这样，我们通过显式转型就可以让一个普通函数成为满足http.Handler接口的类型了…
展开

作者回复: 这三个问题问的都非常棒👍

这里是第一讲，因此很多技术术语可能没有更细致的解释，随着课程的展开，一些术语在后续的课程讲解中都会涉及到。

既然提出这些问题了，我就来简单解答一下。

第一个问题：什么是编译器的前端？在计算机专业编译原理课上大家可能都会学到一个编译器的完整编译过程包括：词法分析，语法分析，类型检查，中间代码生成，代码优化，目标代码生成，目标代码优化等几个阶段。从词法分析到中间代码生成，这就是编译器前端所要负责的事情。而代码优化和目标代码生成，则是编译器后端的职责。

文中提到了实现了一个gcc的前端，就是实现了一个程序，这个程序可以读取go源码并通过词法分析、语法分析、类型检查，最终生成中间代码，而这个中间代码可以被gcc后端所识别并生成最终目标代码。

第二个问题：什么是显式转型？这是一个语法层面的术语。举个简单例子，在Go中，如果一个变量的类型为int，但我要将其与另一个int64类型的变量进行加法运算，我们不能直接将它们相加，我们必须将它们显式的转换为同一类型后才能相加，这里的转换过程就简称为显式转型，这个例子转换为代码就是：

```
var a int = 5
var b int64 = 6
c := int64(a) + b
```

其中的int64(a)就是对变量a进行显式转型。

第三个问题，什么是go运行时？

go 运行时，也称为go runtime。

它在那里？其本身就是每个go程序的一部分，它会跟你的源码一起编译并连接到目标程序中。即便你只是写了一个hello world程序，这个程序中也包含了runtime的实现。

它在我的程序中具体负责什么？runtime负责实现go的垃圾收集、并发、内存堆栈管理以及Go语言的其他关键功能。

它的代码在哪里？它大部分以标准库的形式存放在每个Go发布版的源码中。

感谢对本专栏的支持！

1 18

罗杰

2021-10-14

相比 C/C++ 每个平台的支持问题，Go 的表现简直好太多了。但有个问题，我疑惑了好长好长时间了，希望 Tony 老师解答：printf 能格式化字符串，换行就要手动添加 "\n"，println 又不能格式化字符串。我想知道为什么要这样的设计？在看我来这就是特别反人类的设定，Rust 的 println!("{}", a); 才是符合直觉的。

展开

作者回复: 净给我出难题😂

这个问题我是这么看的，printf是go提供的标准格式化io的函数，它能实现你所期望的所有功能。与c语言的printf是对等的。但println这个函数你可以看成是一种“语法糖”，它本身就是一个特例，你可以用go doc看看println的manual，println原语义就是使用一种默认的格式输出数据到stdout上。你认同这种默认格式，你就使用println，简化你的代码。否则，你就用printf就好了。

5

布凡

2021-10-13

dapr也是go编写

展开

作者回复: 是的，微软牵头推出的云原生分布式应用开发框架dapr的默认实现就是用的Go。

1

陈东

2021-10-15

先掌握概念再慢慢消化。

展开

雪域

2021-10-14

Go后端开发人员，我来深造一下，期待跟随老师能有更大的进步，对go有更深的理解。

作者回复: 感谢对专栏的支持，希望你能得到你期待的收获。加油！

AIK

2021-10-14

内存管理，Go协程，面向对象

展开

aoe

2021-10-14

go有吉祥物，其他2个语言没有

展开

Geek_8ed998

2021-10-13

更新节奏是一周几更啊

展开

编辑回复: 每周一、三、五各更新一讲

西米

2021-10-13

相比c和c++ ，go有了垃圾回收，更快的编译速度

展开

1